

A Appendix

A.1 Free-energy derivation

The following is a derivation of the well-known formula for the free-energy of an RBM. This tractable form is made possible by the bipartite interaction structure of the RBM's units:

$$\begin{aligned}
p(\mathbf{x}) &= \sum_{\mathbf{h}} \frac{1}{Z_\theta} \exp(\mathbf{x}^\top W \mathbf{h} + \mathbf{c}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h}) \\
&= \frac{1}{Z_\theta} \exp(\mathbf{c}^\top \mathbf{x}) \prod_j \sum_{h_j \in \{0,1\}} \exp(\mathbf{x}^\top [W]_j h_j + b_j h_j) \\
&= \frac{1}{Z_\theta} \exp(\mathbf{c}^\top \mathbf{x}) \exp\left(\sum_j (\log \sum_{h_j \in \{0,1\}} \exp(\mathbf{x}^\top [W]_j h_j + b_j h_j))\right) \\
&= \frac{1}{Z_\theta} \exp(\mathbf{c}^\top \mathbf{x} + \sum_j \log[1 + \exp(\mathbf{x}^\top [W]_j + b_j)]) \\
&= \frac{1}{Z_\theta} \exp(-F_\theta(\mathbf{x}))
\end{aligned}$$

A.2 Proofs for Section 2.4

We begin with a useful technical result:

Proposition 11. *For arbitrary $y \in \mathbb{R}$ the following basic facts for the softplus function hold:*

$$\begin{aligned}
y - \text{soft}(y) &= -\text{soft}(-y) \\
\text{soft}(y) &\leq \exp(y)
\end{aligned}$$

Proof. The first fact follows from:

$$\begin{aligned}
y - \text{soft}(y) &= \log(\exp(y)) - \log(1 + \exp(y)) = \log\left(\frac{\exp(y)}{1 + \exp(y)}\right) \\
&= \log\left(\frac{1}{\exp(-y) + 1}\right) = -\log(1 + \exp(y)) = -\text{soft}(-y)
\end{aligned}$$

To prove the second fact, we will show that the function $f(y) = \exp(y) - \text{soft}(y)$ is positive. Note that f tends to 0 as y goes to $-\infty$ since both $\exp(y)$ and $\text{soft}(y)$ do. It remains to show that f is monotonically increasing, which we establish by showing that its derivative is positive:

$$\begin{aligned}
f'(y) &= \exp(y) - \frac{1}{1 + \exp(-y)} > 0 \\
&\Leftrightarrow \exp(y)(1 + \exp(-y)) - \frac{1 + \exp(-y)}{1 + \exp(-y)} > 0 \\
&\Leftrightarrow \exp(y) + 1 - 1 > 0 \quad \Leftrightarrow \exp(y) > 0
\end{aligned}$$

□

Proof of Lemma 2. Consider a single neuron in the RBM network and the corresponding neuron in the hardplus RBM network, whose net-input are given by $y = w^\top x + b$.

For each x , there are two cases for y . If $y \geq 0$, we have by hypothesis that $y \geq C$, and so:

$$\begin{aligned}
|\text{hard}(y) - \text{soft}(y)| &= |y - \text{soft}(y)| = |-\text{soft}(-y)| = \text{soft}(-y) \\
&\leq \exp(-y) \leq \exp(-C)
\end{aligned}$$

And if $y < 0$, we have by hypothesis that $y \leq -C$ and so:

$$\begin{aligned}
|\text{hard}(y) - \text{soft}(y)| &= |0 - \text{soft}(y)| = \text{soft}(y) \\
&\leq \exp(y) \leq \exp(-C)
\end{aligned}$$

Thus, each corresponding pair of neurons computes the same function up to an error bounded by $\exp(-C)$. From this it is easy to show that the entire circuits compute the same function, up to an error bounded by $m \exp(-C)$, as required. \square

Proof of Theorem 3. Suppose we have a softplus RBM network with a number of hidden neurons given by m . To simulate this with a hardplus RBM network we will replace each neuron with a group of hardplus neurons with weights and biases chosen so that the sum of their outputs approximates the output of the original softplus neuron, to within a maximum error of $1/p$ where p is some constant > 0 .

First we describe the construction for the simulation of a single softplus neurons by a group of hardplus neurons.

Let g be a positive integer and $a > 0$. We will define these more precisely later, but for what follows their precise value is not important.

At a high level, this construction works by approximating $\text{soft}(y)$, where y is the input to the neuron, by a piece-wise linear function expressed as the sum of a number of hardplus functions, whose “corners” all lie inside $[-a, a]$. Outside this range of values, we use the fact that $\text{soft}(y)$ converges exponentially fast (in a) to 0 on the left, and y on the right (which can both be trivially computed by hardplus functions).

Formally, for $i = 1, 2, \dots, g, g + 1$, let:

$$q_i = (i - 1) \frac{2a}{g} - a$$

For $i = 1, 2, \dots, g$, let:

$$\nu_i = \frac{\text{soft}(q_{i+1}) - \text{soft}(q_i)}{q_{i+1} - q_i}$$

and also let $\nu_0 = 0$ and $\nu_{g+1} = 1$. Finally, for $i = 1, 2, \dots, g, g + 1$, let:

$$\eta_i = \nu_i - \nu_{i-1}$$

With these definitions it is straightforward to show that $1 \geq \nu_i > 0$, $\nu_i > \nu_{i-1}$ and consequently $0 < \eta_i < 1$ for each i . It is also easy to show that $q_i > q_{i-1}$, $q_0 = -a$ and $q_{g+1} = a$.

For $i = 1, 2, \dots, g, g + 1$, we will set the weight vector \mathbf{w}_i and bias b_i of the i -th hardplus neuron in our group so that the neuron outputs $\text{hard}(\eta_i(y - q_i))$. This is accomplished by taking $\mathbf{w}_i = \eta_i \mathbf{w}$ and $b_i = \eta_i(b - q_i)$, where \mathbf{w} and b (without the subscripts), are the weight vector and bias of the original softplus neuron.

Note that since $|\eta_i| \leq 1$ we have that the weights of these hard neurons are smaller in magnitude than the weights of the original soft neuron and thus bounded by C as required.

The total output (sum) for this group is:

$$T(y) = \sum_{i=1}^{g+1} \text{hard}(\eta_i(y - q_i))$$

We will now bound the approximation error $|T(y) - \text{soft}(y)|$ of our single neuron simulation.

Note that for a given y we have that the i -th hardplus neuron in the group has a non-negative input iff $y \geq q_i$. Thus for $y < -a$ all of the neurons have a negative input. And for $y \geq a$, if we take j to be the largest index i s.t. $q_i \leq y$, then each neuron from $i = 1$ to $i = j$ will have positive input and each neuron from $i = j + 1$ to $i = g + 1$ will have negative input.

Consider the case that $y < -a$. Since the input to each neuron is negative, they each output 0 and thus $T(y) = 0$. This results in an approximation error $\leq \exp(-a)$:

$$|T(y) - \text{soft}(y)| = |0 - \text{soft}(y)| = \text{soft}(y) < \text{soft}(-a) \leq \exp(-a)$$

Next, consider the case that $y \geq -a$, and let j be as given above. In such a case we have:

$$\begin{aligned}
T(y) &= \sum_{i=1}^{g+1} \text{hard}(\eta_i(y - q_i)) = \sum_{i=1}^j \eta_i(y - q_i) + 0 \\
&= \sum_{i=1}^j (\nu_i - \nu_{i-1})(y - q_i) \\
&= y \sum_{i=1}^j (\nu_i - \nu_{i-1}) - \sum_{i=1}^j (\nu_i - \nu_{i-1})q_i \\
&= y\nu_j - y\nu_0 - \nu_j q_j + \sum_{i=1}^{j-1} \nu_i(q_{i+1} - q_i) + \nu_0 q_1 \\
&= \nu_j(y - q_j) + \sum_{i=1}^{j-1} (\text{soft}(q_{i+1}) - \text{soft}(q_i)) \\
&= \nu_j(y - q_j) + \text{soft}(q_j) - \text{soft}(q_1)
\end{aligned}$$

For $y \leq a$ we note that $\nu_j(y - q_j) + \text{soft}(q_j)$ is a secant approximation to $\text{soft}(y)$ generated by the secant from q_j to q_{j+1} and upperbounds $\text{soft}(y)$ for $y \in [q_j, q_{j+1}]$. Thus a crude bound on the error is $\text{soft}(q_{j+1}) - \text{soft}(q_j)$, which only makes use of the fact that $\text{soft}(y)$ is monotonic. Then because the slope (derivative) of $\text{soft}(y)$ is $\sigma(y) = 1/(1 + \exp(-y)) < 1$, we can further (crudely) bound this by $q_{j+1} - q_j$. Thus the approximation error at such y 's may be bounded as:

$$\begin{aligned}
|T(y) - \text{soft}(y)| &= |\nu_j(y - q_j) + \text{soft}(q_j) - \text{soft}(q_1) - \text{soft}(y)| \\
&\leq \max\{|\nu_j(y - q_j) + \text{soft}(q_j) - \text{soft}(y)|, \text{soft}(q_1)\} \\
&\leq \max\{q_{j+1} - q_j, \exp(-a)\} = \max\left\{\frac{2a}{g}, \exp(-a)\right\}
\end{aligned}$$

where we have also used $\text{soft}(q_1) = \text{soft}(-a) \leq \exp(-a)$.

For the case $y > a$, all $q_i > y$ and the largest index j such that $q_j \leq y$ is $j = g + 1$. So $\nu_j(y - q_j) + \text{soft}(q_j) - \text{soft}(q_1) = y - a + \text{soft}(a) - \text{soft}(-a) = y$. Thus the approximation error at such y 's is:

$$|y - \text{soft}(y)| = |- \text{soft}(-y)| = \text{soft}(-y) \leq \text{soft}(-a) \leq \exp(-a)$$

Having covered all cases for y we conclude that the general approximation error for a single softplus neuron satisfies the following bound:

$$|y - \text{soft}(y)| \leq \max\left\{\frac{2a}{g}, \exp(-a)\right\}$$

For a softplus RBM network with m neurons, our hardplus RBM neurons constructed by replacing each neuron with a group of hardplus neurons as described above will require a total of $m(g + 1)$ neurons, and have an approximation error bounded by the sum of the individual approximation errors, which is itself bounded by:

$$m \max\left\{\frac{2a}{g}, \exp(-a)\right\}$$

Taking $a = \log(mp)$, $g = \lceil 2mpa \rceil$. This gives:

$$\begin{aligned}
m \max\left\{\frac{2a}{\lceil 2mpa \rceil}, \frac{1}{mp}\right\} &\leq m \max\left\{\frac{2a}{2mpa}, \frac{1}{mp}\right\} \\
&= \max\left\{\frac{1}{p}, \frac{1}{p}\right\} = \frac{1}{p}
\end{aligned}$$

Thus we see that with $m(g + 1) = m(\lceil 2mp \log(mp) \rceil + 1) \leq 2m^2 p \log(mp) + m$ neurons we can produce a hardplus RBM network which approximates the output of our softplus RBM network with error bounded by $1/p$. \square

Remark 12. Note that the construction used in the above lemma is likely far from optimal, as the placement of the q_i 's could be done more carefully. Also, the error bound we proved is crude and does not make strong use of the properties of the softplus function. Nonetheless, it seems good enough for our purposes.

A.3 Proofs for Section 2.5

Proof of Proposition 5. Suppose that there is an RBM network of size m with weights bounded in magnitude by C computes a function g which represent f with margin δ .

Then taking $p = 2/\delta$ and applying Theorem 3 we have that there exists an hardplus RBM network of size $4m^2 \log(2m/\delta)/\delta + m$ which computes a function g' s.t. $|g(\mathbf{x}) - g'(\mathbf{x})| \leq 1/p = \delta/2$ for all \mathbf{x} .

Note that $f(\mathbf{x}) = 1 \Rightarrow \text{thresh}(g(\mathbf{x})) = 1 \Rightarrow g(\mathbf{x}) \geq \delta \Rightarrow g'(\mathbf{x}) \geq \delta - \delta/2 = \delta/2$ and similarly, $f(\mathbf{x}) = 0 \Rightarrow \text{thresh}(g(\mathbf{x})) = 0 \Rightarrow g(\mathbf{x}) \leq -\delta \Rightarrow g'(\mathbf{x}) \leq -\delta + \delta/2 = -\delta/2$. Thus we conclude that g' represents f with margin $\delta/2$. □

A.4 Proofs for Section 2.7

Proof of Theorem 6. Let f be a Boolean function on n variables computed by a size s hardplus RBM network, with parameters (W, b, d) . We will first construct a three layer hybrid Boolean/threshold circuit/network where the output gate is a simple weighted sum, the middle layer consists of AND gates, and the bottom hidden layer consists of threshold neurons. There will be $n \cdot m$ AND gates, one for every $i \in [n]$ and $j \in [m]$. The $(i, j)^{\text{th}}$ AND gate will have inputs: (1) x_i and (2) $(\mathbf{x}^\top [W]_j \geq b_j)$. The weights going from the $(i, j)^{\text{th}}$ AND gate to the output will be given by $[W]_{i,j}$. It is not hard to see that our three layer network computes the same Boolean function as the original hardplus RBM network.

In order to obtain a single hidden layer threshold network, we replace each sub-network rooted at an AND gate of the middle layer by a single threshold neuron. Consider a general sub-network consisting of an AND of: (1) a variable x_j and (2) a threshold neuron computing $(\sum_{i=1}^n a_i x_i \geq b)$. Let Q be some number greater than the sum of all the a_i 's. We replace this sub-network by a single threshold gate that computes $(\sum_{i=1}^n a_i x_i + Q x_j \geq b + Q)$. Note that if the input x is such that $\sum_i a_i x_i \geq b$ and $x_j = 1$, then $\sum_i a_i x_i + Q x_j$ will be at least $b + Q$, so the threshold gate will output 1. In all other cases, the threshold will output zero. (If $\sum_i a_i x_i < b$, then even if $x_j = 1$, the sum will still be less than $Q + b$. Similarly, if $x_j = 0$, then since $\sum_i a_i x_i$ is never greater than $\sum_i a_i$, the total sum will be less than $Q \leq (n+1)C$.) □

A.5 Proof of Theorem 7

Proof. We will first describe how to construct a hardplus RBM network which satisfies the properties required for part (i). It will be composed of n special groups of hardplus neurons (which are defined and discussed below), and one additional one we call the “zero-neuron”, which will be defined later.

Definition 13 A “building block” is a group of n hardplus neurons, parameterized by the scalars γ and e , where the weight vector $\mathbf{w} \in \mathbb{R}^n$ between the i -th neuron in the group and the input layer is given by $w_i = M - \gamma$ and $w_j = -\gamma$ for $j \neq i$ and the bias will be given by $b = \gamma e - M$, where M is a constant chosen so that $M > \gamma e$.

For a given \mathbf{x} , the input to the i -th neuron of a particular building block is given by:

$$\begin{aligned} \sum_{j=1}^n w_j x_j + b &= w_i x_i + \sum_{j \neq i} w_j x_j + b \\ &= (M - \gamma) x_i - \gamma (X - x_i) + \gamma e - M \\ &= \gamma (e - X) - M(1 - x_i) \end{aligned}$$

When $x_i = 0$, this is $\gamma(e - X) - M < 0$, and so the neuron will output 0 (by definition of the hardplus function). On the other hand, when $x_i = 1$, the input to the neuron will be $\gamma(e - X)$ and thus the output will be $\max(0, \gamma(e - X))$.

In general, we have that the output will be given by:

$$x_i \max(0, \gamma(e - X))$$

From this it follows that the combined output from the neurons in the building block is:

$$\begin{aligned} \sum_{i=1}^n (x_i \max(0, \gamma(e - X))) &= \max(0, \gamma(e - X)) \sum_{i=1}^n x_i \\ &= \max(0, \gamma(e - X))X = \max(0, \gamma X(e - X)) \end{aligned}$$

Note that whenever X is positive, the output is a concave quadratic function in X , with zeros at $X = 0$ and $X = e$, and maximized at $X = e/2$, with value $\gamma e^2/4$.

Next we show how the parameters of the n building blocks used in our construction can be set to produce a hardplus RBM network with the desired output.

First, define d to be any number greater than or equal to $2n^2 \sum_j |t_j|$.

Indexing the building blocks by j for $1 \leq j \leq n$ we define their respective parameters γ_j, e_j as follows:

$$\begin{aligned} \gamma_n &= \frac{t_n + d}{n^2}, & \gamma_j &= \frac{t_j + d}{j^2} - \frac{t_{j+1} + d}{(j+1)^2} \\ e_n &= 2n, & e_j &= \frac{2}{\gamma_j} \left(\frac{t_j + d}{j} - \frac{t_{j+1} + d}{j+1} \right) \end{aligned}$$

where we have assumed that $\gamma_j \neq 0$ (which will be established, along with some other properties of these definitions, in the next claim).

Claim 1. For all j , $1 \leq j \leq n$, (i) $\gamma_j > 0$ and (ii) for all j , $1 \leq j \leq n-1$, $j \leq e_j \leq j+1$.

Proof of Claim 1. Part (i): For $j = n$, by definition we know that $\gamma_n = \frac{t_n + d}{n^2}$. For $d \geq 2n^2 \sum_j |t_j| > |t_n|$, the numerator will be positive and therefore γ_n will be positive.

For $j < n$, we have:

$$\begin{aligned} &\gamma_j > 0 \\ \Leftrightarrow &\frac{t_j + d}{j^2} > \frac{t_{j+1} + d}{(j+1)^2} \\ \Leftrightarrow &(j+1)^2(t_j + d) > j^2(t_{j+1} + d) \\ \Leftrightarrow &d((j+1)^2 - j^2) > j^2 t_{j+1} - (j+1)^2 t_j \\ \Leftrightarrow &d > \frac{j^2 t_{j+1} - (j+1)^2 t_j}{2j+1} \end{aligned}$$

The right side of the above inequality is less than or equal to $\frac{(j+1)^2(|t_{j+1}| + |t_j|)}{2j+1} \leq (j+1)(|t_{j+1}| + |t_j|)$ which is strictly upper bounded by $2n^2 \sum_j |t_j|$, and thus by d . So it follows that $\gamma_j > 0$ as needed.

Part (ii):

$$\begin{aligned}
j &\leq e_j = \frac{2}{\gamma_j} \left(\frac{t_j + d}{j} - \frac{t_{j+1} + d}{j+1} \right) \\
\Leftrightarrow j\gamma_j &\leq 2 \left(\frac{t_j + d}{j} - \frac{t_{j+1} + d}{j+1} \right) \\
\Leftrightarrow \frac{t_j + d}{j} - \frac{j(t_{j+1} + d)}{(j+1)^2} &\leq 2 \left(\frac{t_j + d}{j} - \frac{t_{j+1} + d}{j+1} \right) \\
\Leftrightarrow -\frac{j(t_{j+1} + d)}{(j+1)^2} &\leq \frac{t_j + d}{j} - 2\frac{t_{j+1} + d}{j+1} \\
\Leftrightarrow -(t_{j+1} + d)j^2 &\leq (t_j + d)(j+1)^2 - 2(t_{j+1} + d)j(j+1) \\
\Leftrightarrow d(j^2 - 2j(j+1) + (j+1)^2) &\geq -j^2t_{j+1} + 2j(j+1)t_{j+1} - (j+1)^2t_j \\
\Leftrightarrow d &\geq -j^2t_{j+1} + 2j(j+1)t_{j+1} - (j+1)^2t_j
\end{aligned}$$

where we have used $j^2 - 2j(j+1) + (j+1)^2 = (j - (j+1))^2 = 1^2 = 1$ at the last line. Thus it suffices to make d large enough to ensure that $j \leq e_j$. For our choice of d , this will be true.

For the upper bound we have:

$$\begin{aligned}
\frac{2}{\gamma_j} \left(\frac{t_j + d}{j} - \frac{t_{j+1} + d}{j+1} \right) &= e_j \leq j+1 \\
\Leftrightarrow 2 \left(\frac{t_j + d}{j} - \frac{t_{j+1} + d}{j+1} \right) &\leq (j+1)\gamma_j = \frac{(j+1)(t_j + d)}{j^2} - \frac{t_{j+1} + d}{j+1} \\
\Leftrightarrow 2\frac{t_j + d}{j} - \frac{t_{j+1} + d}{j+1} &\leq \frac{(j+1)(t_j + d)}{j^2} \\
\Leftrightarrow 2(t_j + d)j(j+1) - (t_{j+1} + d)j^2 &\leq (j+1)^2(t_j + d) \\
\Leftrightarrow \frac{-(d - t_{j+1})}{j+1} + 2\frac{(d + t_j)}{j} &\leq (j+1)\frac{(d + t_j)}{j^2} \\
\Leftrightarrow -j^2(d + t_{j+1}) + 2j(j+1)(d + t_j) &\leq (j+1)^2(d + t_j) \\
\Leftrightarrow d(j^2 - 2j(j+1) + (j+1)^2) &\geq -j^2t_{j+1} + 2j(j+1)t_j - (j+1)^2t_j \\
\Leftrightarrow d &\geq -j^2t_{j+1} + 2j(j+1)t_j - (j+1)^2t_j
\end{aligned}$$

where we have used $j^2 - 2j(j+1) + (j+1)^2 = 1$ at the last line. Again, for our choice of d the above inequality is satisfied. \square

Finally, define M to be any number greater than $\max(t_0 + d, \max_i\{\gamma_i e_i\})$.

In addition to the n building blocks, our hardplus RBM will include an addition unit that we will call the *zero*-neuron, which handles $\mathbf{x} = \mathbf{0}$. The zero-neuron will have weights \mathbf{w} defined by $w_i = -M$ for each i , and $b = t_0 + d$.

Finally, the output bias B of our hardplus RBM network will be set to $-d$.

The total output of the network is simply the sum of the outputs of the n different building blocks, the zero neuron, and constant bias $-d$.

To show part (i) of the theorem we want to prove that for all k , whenever $X = k$, our circuit outputs the value t_k .

We make the following definitions:

$$a_k \equiv -\sum_{j=k}^n \gamma_j \quad b_k \equiv \sum_{j=k}^n \gamma_j e_j$$

Claim 2.

$$a_k = \frac{-(t_k + d)}{k^2} \quad b_k = \frac{2(t_k + d)}{k} \quad b_k = -2ka_k$$

This claim is self-evidently true by examining basic definitions of γ_j and e_j and realizing that a_k and b_k are telescoping sums.

Given these facts, we can prove the following:

Claim 3. For all k , $1 \leq k \leq n$, when $X = k$ the sum of the outputs of all the n building blocks is given by $t_k + d$.

Proof of Claim 3. For $X = n$, the (γ_n, e_n) -block computes $\max(0, \gamma_n X(e_n - X)) = \max(0, -\gamma_n X^2 + \gamma_n e_n X)$. By the definition of e_n , $n \leq e_n$, and thus when $X \leq n$, $\gamma_n X(e_n - X) \geq 0$. For all other building blocks (γ_j, e_j) , $j < n$, since $e_j \leq j + 1$, this block outputs zero since $\gamma_j X(e_j - X)$ is less than or equal to zero. Thus the sum of all of the building blocks when $X = n$ is just the output of the (γ_n, e_n) -block which is

$$\gamma_n \cdot n(e_n - n) = -\gamma_n \cdot n^2 + \gamma_n e_n \cdot n = -(t_n + d) + 2(t_n + d) = t_n + d$$

as desired.

For $X = k$, $1 \leq k < n$ the argument is similar. For all building blocks $j \geq k$, by Claim 1 we know that $e_j \geq j$ and therefore this block on $X = k$ is nonnegative and therefore contributes to the sum. On the other hand, for all building blocks $j < k$, by Claim 1 we know that $e_j \leq j + 1$ and therefore this outputs 0 and so does not contribute to the sum.

Thus the sum of all of the building blocks is equal to the sum of the non-zero regions of the building blocks j for $j \geq k$. Since each of this is a quadratic function of X , it can be written as a single quadratic polynomial of the form $a_k X^2 + b_k X$ where a_k and b_k are defined as before.

Plugging in the above expressions for a_k and b_k from Claim 2, we see that the value of this polynomial at $X = k$ is:

$$a_k k^2 + b_k k = \frac{-(t_k + d)}{k^2} k^2 + \frac{2(t_k + d)}{k} k = -(t_k + d) + 2(t_k + d) = t_k + d$$

□

Finally, it remains to ensure that our hardplus RBM network outputs t_0 for $X = 0$. Note that the sum of the outputs of all n building blocks and the output bias is $-d$ at $X = 0$. To correct this, we set the incoming weights and the bias of the zero-neuron according to $w_i = -M$ for each i , and $b = t_0 + d$. When $X = 0$, this neuron will output $t_0 + d$, making the total output of the network $-d + t_0 + d = t_0$ as needed. Furthermore, note that the addition of the zero-neuron does not affect the output of the network when $X = k > 0$ because the zero-neuron outputs 0 on all of these inputs as long as $M \geq t_0 + d$.

This completes the proof of part (i) of the theorem and it remains to prove part (ii).

Observe that the size of the weights grows linearly in M and d , which follows directly from their definitions. And note that the magnitude of the input to each neuron is lower bounded by a positive linear function of M and d (a non-trivial fact which we will prove below). From these two observations it follows that to achieve the condition that the magnitude of the input to each neuron is greater than $C(n)$ for some function C of n , the weights need to grow linearly with C . Noting that error bound condition $\epsilon \leq (n^2 + 1) \exp(-C)$ in Lemma 2 can be rewritten as $C \leq \log((n^2 + 1)) + \log(1/\epsilon)$, from which part (ii) of the theorem then follows.

There are two cases where a hardplus neuron in building block j has a negative input. Either the input is $\gamma_j(e_j - X) - M$, or it is $\gamma_j(e_j - X)$ for $X \geq j + 1$. In the first case it is clear that as M grows the net input becomes more negative since e_j doesn't depend on M at all.

The second case requires more work. First note that from its definition, e_j can be rewritten as $2 \frac{(j+1)a_{j+1} - ja_j}{\gamma_j}$. Then for any $X \geq j+1$ and $j \leq n-1$ we have:

$$\begin{aligned}
\gamma_j(e_j - X) &\leq \gamma_j(e_j - (j+1)) \\
&= \gamma_j \left(2 \frac{(j+1)a_{j+1} - ja_j}{\gamma_j} - (j+1) \right) \\
&= 2(j+1)a_{j+1} - 2ja_j - (j+1)\gamma_j \\
&= 2(j+1)a_{j+1} - 2ja_j - (j+1)(a_{j+1} - a_j) \\
&= (j+1)a_{j+1} - 2ja_j + (j+1)a_j \\
&= \frac{-(d - t_{j+1})}{j+1} + 2 \frac{(d + t_{j+1})}{j} - (j+1) \frac{d + t_{j+1}}{j^2} \\
&= \frac{-j^2(d + t_{j+1}) + 2j(j+1)(d + t_j) - (j+1)^2(d + t_j)}{j^2(j+1)} \\
&= \frac{-(j^2 - 2j(j+1) + (j+1)^2)d - j^2t_j + 2j(j+1)t_j}{j^2(j+1)} \\
&= \frac{-(j - (j+1))^2d - j^2t_j + 2j(j+1)t_j}{j^2(j+1)} \\
&= \frac{-d - j^2t_j + 2j(j+1)t_j}{j^2(j+1)} \\
&= \frac{-d}{j^2(j+1)} + \frac{-j^2t_j + 2j(j+1)t_j}{j^2(j+1)}
\end{aligned}$$

So we see that as d increases, this bound guarantees that $\gamma_j(e_j - X)$ becomes more negative for each $X \geq j+1$. Also note that for the special zero-neuron, for $X \geq 1$ the net input will be $-MX + t_0 + d \leq -M + t_0 + d$, which will shrink as M grows.

For neurons belonging to building block j which have a positive valued input, we have that $X < e_j$. Note that for any $X \leq j$ and $j < n$ we have:

$$\begin{aligned}
\gamma_j(e_j - X) &\geq \gamma_j(e_j - j) = \gamma_j \left(2 \frac{(j+1)a_{j+1} - ja_j}{\gamma_j} - j \right) \\
&= 2(j+1)a_{j+1} - 2ja_j - j\gamma_j \\
&= 2(j+1)a_{j+1} - 2ja_j - j(a_{j+1} - a_j) \\
&= 2(j+1)a_{j+1} - ja_j - ja_{j+1} \\
&= 2 \frac{-(d + t_{j+1})}{j+1} + \frac{(d + t_j)}{j} + j \frac{(d + t_{j+1})}{(j+1)^2} \\
&= \frac{-2j(j+1)(d + t_{j+1}) + (j+1)^2(d + t_j) + j^2(d + t_{j+1})}{j(j+1)^2} \\
&= \frac{((j+1)^2 - 2j(j+1) + j^2)d + (j+1)^2t_j - 2j(j+1)t_{j+1} + j^2t_{j+1}}{j(j+1)^2} \\
&= \frac{(j+1-j)^2d + (j+1)^2t_j - 2j(j+1)t_{j+1} + j^2t_{j+1}}{j(j+1)^2} \\
&= \frac{d + (j+1)^2t_j - 2j(j+1)t_{j+1} + j^2t_{j+1}}{j(j+1)^2} \\
&= \frac{d}{j(j+1)^2} + \frac{(j+1)^2t_j - 2j(j+1)t_{j+1} + j^2t_{j+1}}{j(j+1)^2}
\end{aligned}$$

And for the case $j = n$, we have for $X \leq j$ that:

$$\gamma_j(e_j - X) \geq \gamma_j(e_j - j) = \frac{d + t_n}{n^2}(2n - n) = \frac{d}{n} + \frac{t_n}{n}$$

So in all cases we see that as d increases, this bound guarantees that $\gamma_j(e_j - X)$ grows linearly. Also note that for the special zero-neuron, the net input will be $t_0 + d$ for $X = 0$, which will grow linearly as d increases. \square

A.6 Proofs for Section 4

A.6.1 Proof of Theorem 8

We first state some basic facts which we need.

Fact 14 (Muroga (1971)). *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function computed by a threshold neuron with arbitrary real incoming weights and bias. There exists a constant K and another threshold neuron computing f , all of whose incoming weights and bias are integers with magnitude at most $2^{Kn \log n}$.*

A direct consequence of the above fact is the following fact, by now folklore, whose simple proof we present for the sake of completeness.

Fact 15. *Let f_n be the set of all Boolean functions on $\{0, 1\}^n$. For each $0 < \alpha < 1$, let $f_{\alpha, n}$ be the subset of such Boolean functions that are computable by threshold networks with one hidden layer with at most s neurons. Then, there exists a constant K such that,*

$$|f_{\alpha, n}| \leq 2^{K(n^2 s \log n + s^2 \log s)}.$$

Proof. Let s be the number of hidden neurons in our threshold network. By using Fact 14 repeatedly for each of the hidden neurons, we obtain another threshold network having still s hidden units computing the same Boolean function such that the incoming weights and biases of all hidden neurons is bounded by $2^{Kn \log n}$. Finally applying Fact 14 to the output neuron, we convert it to a threshold gate with parameters bounded by $2^{Ks \log s}$. Henceforth, we count only the total number of Boolean functions that can be computed by such threshold networks with integer weights. We do this by establishing a simple upper bound on the total number of distinct such networks. Clearly, there are at most $2^{Kn^2 \log n}$ ways to choose the incoming weights of a given neuron in the hidden layer. There are s incoming weights to choose for the output threshold, each of which is an integer of magnitude at most $2^{Ks \log s}$. Combining these observations, there are at most $2^{Ks \cdot n^2 \log n} \times 2^{Ks^2 \log s}$ distinct networks. Hence, the total number of distinct Boolean functions that can be computed is at most $2^{K(n^2 s \log n + s^2 \log s)}$. \square

With these basic facts in hand, we prove below Theorem 8 using Proposition 5 and Theorem 6.

Proof of Theorem 8. Consider any thresholded RBM network with m hidden units that is computing a n -dimensional Boolean function with margin δ . Using Proposition 5, we can obtain a thresholded hardplus RBM network of size $4m^2/\delta \cdot \log(2m/\delta) + m$ that computes the same Boolean function as the thresholded original RBM network. Applying Theorem 6 and thresholding the output, we obtain a thresholded network with 1 hidden layer of thresholds which is the same size and computes the same Boolean function. This argument shows that the set of Boolean functions computed by thresholded RBM networks of m hidden units and margin δ is a subset of the Boolean functions computed by 1-hidden-layer threshold networks of size $4m^2n/\delta \cdot \log(2m/\delta) + mn$. Hence, invoking Fact 15 establishes our theorem. \square

A.6.2 Proof of Theorem 9

Note that the theorems from Hajnal et al. (1993) assume integer weights, but this hypothesis can be easily removed from their Theorem 3.6. In particular, Theorem 3.6 assumes nothing about the lower weights, and as we will see, the integrality assumption on the top level weights can be easily replaced with a margin condition.

First note that their Lemma 3.3 only uses the integrality of the upper weights to establish that the margin must be ≥ 1 . Otherwise it is easy to see that with a margin δ , Lemma 3.3 implies that a threshold neuron in a thresholded network of size m is a $\frac{2\delta}{\alpha}$ -discriminator (α is the sum of the

absolute values of the 2nd-level weights in their notation). Then Theorem 3.6's proof gives $m \geq \delta 2^{(1/3-\epsilon)n}$ for sufficiently large n (instead of just $m \geq 2^{(1/3-\epsilon)n}$). A more precise bound that they implicitly prove in Theorem 3.6 is $m \geq \frac{6\delta 2^{n/3}}{C}$.

Thus we have the following fact adapted from Hajnal et al. (1993):

Fact 16. *For a neural network of size m with a single hidden layer of threshold neurons and weights bounded by C that computes a function that represents IP with margin δ , we have $m \geq \frac{6\delta 2^{n/3}}{C}$.*

Proof of Theorem 9. By Proposition 5 it suffices to show that no thresholded hardplus RBM network of size $\leq 4m^2 \log(2m/\delta)/\delta + m$ with parameters bounded by C can compute IP with margin $\delta/2$.

Well, suppose by contradiction that such a thresholded RBM network exists. Then by Theorem 6 there exists a single hidden layer threshold network of size $\leq 4m^2 n \log(2m/\delta)/\delta + mn$ with weights bounded in magnitude by $(n+1)C$ that computes the same function, i.e. one which represents IP with margin $\delta/2$.

Applying the above Fact we have $4m^2 n \log(2m/\delta)/\delta + mn \geq \frac{3\delta 2^{n/3}}{(n+1)C}$.

It is simple to check that this bound is violated if m is bounded as in the statement of this theorem. □

A.6.3 Proof of Theorem 10

We prove a more general result here from which we easily derive Theorem 10 as a special case. To state this general result, we introduce some simple notions. Let $h : \mathbb{R} \rightarrow \mathbb{R}$ be an *activation function*. We say h is monotone if it satisfies the following: Either $h(x) \leq h(y)$ for all $x < y$ OR $h(x) \geq h(y)$ for all $x < y$. Let $\ell : \{0, 1\}^n \rightarrow \mathbb{R}$ be an inner function. An (h, ℓ) gate/neuron $G_{h,\ell}$ is just one that is obtained by composing h and ℓ in the natural way, i.e. $G_{h,\ell}(x) = h(\ell(x))$. We notate $\|(h, \ell)\|_\infty = \max_{x \in \{0,1\}^n} |G_{h,\ell}(x)|$.

We assume for the discussion here that the number of input variables (or observables) is even and is divided into two halves, called x and y , each being a Boolean string of n bits. In this language, the inner production Boolean function, denoted by $IP(x, y)$, is just defined as $x_1 y_1 + \dots + x_n y_n \pmod{2}$. We call an inner function of a neuron/gate to be (x, y) -separable if it can be expressed as $g(x) + f(y)$. For instance, all affine inner functions are (x, y) -separable. Finally, given a set of activation functions H and a set of inner functions I , an (H, I) -network is one each of whose hidden unit is a neuron of the form $G_{h,\ell}$ for some $h \in H$ and $\ell \in I$. Let $\|(H, I)\|_\infty = \sup \{ \|(h, \ell)\|_\infty : h \in H, \ell \in I \}$.

Theorem 17. *Let H be any set of monotone activation functions and I be a set of (x, y) separable inner functions. Then, every (H, I) network with one layer of m hidden units computing IP with a margin of δ must satisfy the following:*

$$m \geq \frac{\delta}{2 \|(H, I)\|_\infty} 2^{n/4}.$$

In order to prove Theorem 17, it would be convenient to consider the following 1/-1 valued function: $(-1)^{IP(x,y)} = (-1)^{x_1 y_1 + \dots + x_n y_n}$. Please note that when IP evaluates to 0, $(-1)^{IP}$ evaluates to 1 and when IP evaluates to 1, $(-1)^{IP}$ evaluates to -1.

We also consider a matrix M_n with entries in $\{1, -1\}$ which has 2^n rows and 2^n columns. Each row of M_n is indexed by a unique Boolean string in $\{0, 1\}^n$. The columns of the matrix are also indexed similarly. The entry $M_n[x, y]$ is just the 1/-1 value of $(-1)^{IP(x,y)}$. We need the following fact that is a special case of the classical result of Lindsey.

Lemma 18 (Chor and Goldreich, 1988). *The magnitude of the sum of elements in every $r \times s$ sub-matrix of M_n is at most $\sqrt{rs} 2^{n/2}$.*

We use Lemma 18 to prove the following key fact about monotone activation functions:

Lemma 19. *Let $G_{h,\ell}$ be any neuron with a monotone activation function h and inner function ℓ that is (x, y) -separable. Then,*

$$\left| \mathbb{E}_{x,y} \left[G_{h,\ell}(x,y) (-1)^{\text{IP}(x,y)} \right] \right| \leq \|(h,\ell)\|_\infty \cdot 2^{-\Omega(n)}. \quad (2)$$

Proof. Let $\ell(x,y) = g(x) + f(y)$ and let $0 < \alpha < 1$ be some constant specified later. Define a total order \prec_g on $\{0,1\}^n$ by setting $x \prec_g x'$ whenever $g(x) \leq g(x')$ and x occurs before x' in the lexicographic ordering. We divide $\{0,1\}^n$ into $t = 2^{(1-\alpha)n}$ groups of equal size as follows: the first group contains the first $2^{\alpha n}$ elements in the order specified by \prec_g , the second group has the next $2^{\alpha n}$ elements and so on. The i th such group is denoted by X_i for $i \leq 2^{(1-\alpha)n}$. Likewise, we define the total order \prec_f and use it to define equal sized blocks $Y_1, \dots, Y_{2^{(1-\alpha)n}}$.

The way we estimate the LHS of (2) is to pair points in the block (X_i, Y_j) with (X_{i+1}, Y_{j+1}) in the following manner: wlog assume that the activation function h is non-decreasing. Then, $G_{h,\ell}(x,y) \leq G_{h,\ell}(x',y')$ for each $(x,y) \in (X_i, Y_j)$ and $(x',y') \in (X_{i+1}, Y_{j+1})$. Further, applying Lemma 18, we will argue that the total number of points in (X_i, Y_j) at which the product in the LHS evaluates negative (positive) is very close to the number of points in (X_{i+1}, Y_{j+1}) at which the product evaluates to positive (negative). Moreover, by assumption, the composed function (h,ℓ) does not take very large values in our domain by assumption. These observations will be used to show that the points in blocks that are diagonally across each other will almost cancel each other's contribution to the LHS. There are too few uncanceled blocks and hence the sum in the LHS will be small. Forthwith the details.

Let $P_{i,j}^+ = \{(x,y) \in (X_i, Y_j) \mid \text{IP}(x,y) = 1\}$ and $P_{i,j}^- = \{(x,y) \in (X_i, Y_j) \mid \text{IP}(x,y) = -1\}$. Let $t = 2^{(1-\alpha)n}$. Let $h_{i,j}$ be the max value that the gate takes on points in (X_i, Y_j) . Note that the non-decreasing assumption on h implies that $h_{i,j} \leq h_{i+1,j+1}$. Using this observation, we get the following:

$$\mathbb{E}_{x,y} \left[G_{h,\ell}(x,y) (-1)^{\text{IP}(x,y)} \right] \leq \frac{1}{4^n} \left| \sum_{(i,j) < t} h_{i,j} \left(|P_{i,j}^+| - |P_{i+1,j+1}^-| \right) \right| + \frac{1}{4^n} \sum_{i=t \text{ OR } j=t} h_{i,j} |P_{i,j}| \quad (3)$$

We apply Lemma 18 to conclude that $|P_{i+1,j+1}^+| - |P_{i,j}^-|$ is at most $2 \cdot 2^{(\alpha+1/2)n}$. Thus, we get

$$\text{RHS of (3)} \leq \|(h,\ell)\|_\infty \cdot (2 \cdot 2^{-(\alpha-\frac{1}{2})n} + 4 \cdot 2^{-(1-\alpha)n}). \quad (4)$$

Thus, setting $\alpha = 3/4$ gives us the bound that the RHS above is arbitrarily close to $\|(h,\ell)\|_\infty \cdot 2^{-n/4}$.

Similarly, pairing things slightly differently, we get

$$\mathbb{E}_{x,y} \left[G_{h,\ell}(x,y) (-1)^{\text{IP}(x,y)} \right] \geq \frac{1}{4^n} \sum_{(i,j) < t} h_{i+1,j+1} \left(|P_{i+1,j+1}^+| - |P_{i,j}^-| \right) - \frac{1}{4^n} \sum_{i=t \text{ OR } j=t} |h_{i,j}| \cdot |P_{i,j}| \quad (5)$$

Again similar conditions and settings of α imply that RHS of (5) is no smaller than $-\|(h,\ell)\|_\infty \cdot 2^{-n/4}$, thus proving our lemma. \square

We are now ready to prove Theorem 17.

Proof of Theorem 17. Let C be any (H,I) network having m hidden units, $G_{h_1,\ell_1}, \dots, G_{h_m,\ell_m}$, where each $h_i \in H$ and each $\ell_i \in I$ is (x,y) -separable. Further, let the output threshold gate be such that whenever the sum is at least b , C outputs 1 and whenever it is at most a , C outputs -1. Then, let f be the sum total of the function feeding into the top threshold gate of C . Define $t = f - (a+b)/2$. Hence,

$$\begin{aligned} \mathbb{E}_{x,y} [f(x,y) (-1)^{\text{IP}(x,y)}] &= \mathbb{E}_{x,y} [t(x,y) (-1)^{\text{IP}(x,y)}] + \frac{a+b}{2} \mathbb{E}_{x,y} [(-1)^{\text{IP}(x,y)}] \\ &\geq (b-a)/2 + \frac{a+b}{2} \mathbb{E}_{x,y} [(-1)^{\text{IP}(x,y)}]. \end{aligned}$$

Thus, it follows easily

$$\left| \mathbb{E}_{x,y} [f(x,y)(-1)^{\text{IP}(x,y)}] \right| \geq \frac{b-a}{2} - \frac{|a+b|}{2} 2^{-n}. \quad (6)$$

On the other hand, by linearity of expectation and applying Lemma 19, we get

$$\left| \mathbb{E}_{x,y} [f(x,y)(-1)^{\text{IP}(x,y)}] \right| \leq \sum_{j=1}^m \left| \mathbb{E}_{x,y} \left[G_{h_j, \ell_j}(x,y)(-1)^{\text{IP}(x,y)} \right] \right| \leq m \cdot \|(H, I)\|_{\infty} \cdot 2^{-n/4}. \quad (7)$$

Comparing (6) and (7), observing that each of $|a|$ and $|b|$ is at most $m \|(H, I)\|_{\infty}$ and recalling that $\delta = (b-a)$, our desired bound on m follows. \square

Proof of Theorem 10. The proof follows quite simply by noting that the set of activation functions in this case is just the singleton set having only the monotone function $\text{soft}(y) = \log(1 + \exp(y))$. The set of inner functions are all affine functions with each coefficient having value at most C . As the affine functions are (x, y) -separable, we can apply Theorem 17. We do so by noting $\|(H, I)\|_{\infty} \leq \log(1 + \exp(nC)) \leq \max\{\log 2, nC + \log 2\}$. That yields our result. \square

Remark 20. *It is also interesting to note that Theorem 17 appears to be tight in the sense that none of the hypotheses can be removed. That is, for neurons with general non-monotonic activation functions, or for neurons with monotonic activation functions whose output magnitude violates the aforementioned bounds, there are example networks that can efficiently compute any real-valued function. Thus, to improve this result (e.g. removing the weight bounds) it appears one would need to use a stronger property of the particular activation function than monotonicity.*